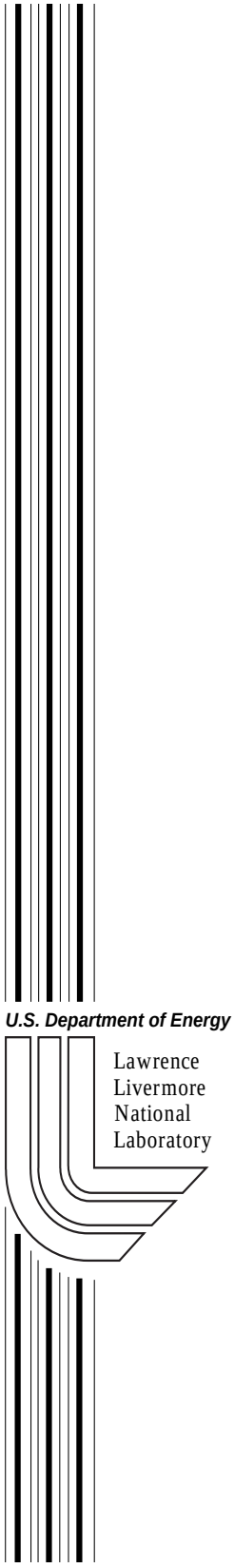
Report: UCRL-ID-126455, Rev. 4
Code Release: UCRL-CODE-2003-043



TART 2002
A Coupled Neutron-Photon 3-D, Combinatorial Geometry Time Dependent Monte Carlo Transport Code

by
Dermott E. Cullen
University of California
Lawrence Livermore National Laboratory
P.O. Box 808
L-159
Livermore, CA 94550

tele: 925-423-7359
e. mail: cullen1@llnl.gov
website: http://www.llnl.gov/cullen1

June 6, 2003

Report: UCRL-ID-126455, Rev. 4
Code Release: UCRL-CODE-2003-043

TART 2002
A Coupled Neutron-Photon
3-D, Combinatorial Geometry
Time Dependent
Monte Carlo Transport Code

by
Dermott E. Cullen
University of California
Lawrence Livermore National Laboratory
P.O. Box 808
L-159
Livermore, CA 94550

tele: 925-423-7359
e. mail: cullen1@llnl.gov
website: http://www.llnl.gov/culen1

June 6, 2003

Abstract

TART 2002 is a coupled neutron-photon, 3 Dimensional, combinatorial geometry, time dependent Monte Carlo radiation transport code. This code can run on **any modern computer**. It is a **complete system** to assist you with input preparation, running Monte Carlo calculations, and analysis of output results. TART 2002 is also **incredibly FAST**; if you have used similar codes, you will be amazed at how fast this code is compared to other similar codes. Use of the entire system can save you a great deal of time and energy.

TART 2002 is distributed on CD. This CD contains on-line documentation for all codes included in the system, the codes configured to run on a variety of computers, and many example problems that you can use to familiarize yourself with the system.

TART 2002 completely supersedes all older versions of TART, and it is strongly recommended that users only use the most recent version of TART 2002 and its data files.

Dedication

This report is dedicated to **Robert J. Howerton**, who recently passed away. For many years Bob led Livermore's Nuclear Data and Monte Carlo efforts. Bob retired in 1986, but continued coming to work until recently, when he finally really retired. Bob was known internationally as well as at the lab for the excellence of his work. Many of our ever changing applications would not have been possible without his efforts to promptly produce nuclear data to meet our needs. Bob will be very much missed both as a co-worker and a friend.

Acknowledgments

I thank the many users of earlier versions of TART who have supplied extremely useful feedback to me. Since the general release of TART, in July 1996, the response from users in terms of feedback has been extremely useful in improving the code. These improvements have been in terms of correcting problems in the earlier releases of TART, and in terms of users proposing new or improved options to meet their needs, now incorporated in TART 2002. I highly encourage all users to supply their feedback to me.

The TART 2002 System

This report is intended merely as a brief introduction to TART 2002. The on-line documentation for the TART 2002 system codes, distributed on TART 2002 CD, has been coordinated to illustrate combined use of the codes to make your job simpler and your work easier to accomplish, in particular extensive use of interactive graphics. If you have not used interactive graphics before you are only making your job harder and your tasks will take longer to accomplish.

Overview of This Report

This report describes all major changes in TART since TART95. As such this report supersedes the reports of TART96 [2], TART97, TART98, and TART2000 [8]. However, the large TART95 report [1] is still the most comprehensive report on TART.

This report is divided into a number of parts, with each part describing one part of the TART 2002 CD system. The parts are,
Part 1: TART 2002 - Monte Carlo Calculations
Part 2: Input Parameters - Description of all Input Parameters
Part 3: TARTCHEK - Check TART Input and Display TART Results
Part 4: TARTAID - Create TART Input
Part 5: EPICSHOW - Display Atomic and Nuclear data used by TART
Part 6: PLOTTAB - General Plotting Code to Display TART Output
Part 7: EDITOR - Text editor for use with TART
Part 8: Utility Codes - A collection of Useful Codes
Part 9: IMAGES – Using TART for medical and industrial imaging

I Strongly Recommend that you read the on-line documentation for all parts of this system, to get a better overall picture of how this entire code system fits together and can help you. The TART 2002 on-line documentation is available in Microsoft Word and Adobe PDF formats, and includes black and white as well as color graphic results. Only when you start using the codes in combination will you realize that this is a complete system that can really assist you in your work.

Computer Requirements

TART 2002 will run on any modern Computer, with at Least 8 Megabytes Memory and 30 Megabytes Disk Space. This puppy can run on virtually any computer; see, the below table of running times on a variety of computers.

TART 2002 CD

TART 2002 is distributed on CD. This CD contains on-line documentation for all codes included in the system, the codes configured to run on a variety of computers, and many example problems that you can use to familiarize yourself with the system.

TART Home Page

The TART home page is now located on the web at,

http://www.llnl.gov/cullen1

This site contains all of the TART documentation, as well as information and documentation related to TART and the nuclear and atomic data that it uses. This site is periodically updated, with reports, tutorials, newsletters, etc. If you are a TART user you should periodically check this site for the latest news.

TART Hot Line

Well, not exactly a hot line, but at least a place to turn to when you need help. If you have any difficulties setting up TART input, running it, or analyzing output, you can contact me at,

Telephone: 925-423-7359
E. Mail: cullen1@llnl.gov

Background: Legacy, Schmegacy

Because TART has a history of over forty years one might think that today's TART code is an old legacy code that is very much out-of-date and is teetering on its last legs. Nothing could be further from the truth. Today's TART code has been completely re-written in the last few years, and is now one of the most modern, up-to-date Monte Carlo

radiation transport codes available. All that remains of the earlier code, **TARTND**, is the user input and output interface, and the hundreds of man-years of experience accumulated in using TART over the last forty years.

The earlier **TARTND** code was written in the LRLTRAN language (unique to Livermore), which limited the code for use only on Livermore's CRAY computers. TARTND was basically a high energy neutron and photon transport code. It was limited to high energy neutrons because it used unshielded, multi-group cross sections, and as such did not account for resonance self-shielding, and it had a very limited thermal scattering treatment, and only a hand full of groups at low energy; none below a milli-eV. In addition its use of Livermore's Evaluated Nuclear Data Library (ENDL) for neutron data further restricted TARTND to high energy neutron problems; as the author of ENDL, Robert J. Howerton stated [11] that ENDL was never intended for use in slow neutron applications, and gives poor results if it is misused in slow neutron applications. In addition TARTND was designed to run on the small computers that were available decades ago, e.g., geometry that could be modeled only in fairly limited detail.

In contrast today's TART code is written in such standard FORTRAN that it runs on everything from large central computers, to UNIX workstations, to MACs and PCs, using Windows or LINUX – like I said: this puppy runs on any computer. It also includes the multi-band method to handle resonance self-shielding [7], and a very accuracy and efficient method to handle free atom scattering [4], as well as thermal scattering law data to handle bound atom scattering [10]. Whereas the old TARTND code used 176 neutron energy groups, mostly clustered at high energy, today's TART code uses 700 neutron energy groups, 50 per energy decade, uniformly spaced in the log of the neutron energy between 10^{-5} eV and 1 GeV. Compared to the hand full of low energy groups used by TARTND, today's TART code has 250 groups below 1 eV. In addition TART now uses the most up-to-date evaluated nuclear data, namely the ENDF/B-VI, Release 8 data [10]. It also uses the most modern evaluated photon interaction data, EPDL97 [6], which is now the official ENDF/B-VI photon interaction data library. In addition today's code is designed to run on today's computers – and tomorrow's computers; there is now no limit on any input parameters, e.g., geometry can be modeled using as much detail as needed, be it using one spatial zone, or a million spatial zones, TART 2002 can efficiently handle both extremes. In an attempt to try and make your life simpler when you do include more geometric details, you will find that TART 2002 now includes many more input options for geometry.

One thing that today's code has managed to carryover from the older TARTND code, is incredible speed. If you compare TART to other Monte Carlo radiation transport codes you will be amazed at how much faster TART is. But be assured that this speed is not accomplished by compromising accuracy; we have always used Howerton's first theorem, "We are in no rush for the wrong answer". The combination of speed and accuracy is achieved by using all appropriate methods that we learned in graduate school and beyond. For example, where other codes may ignore everything we learned and try to solve transport problems starting from basic principles, TART uses very efficient and accurate methods to account for resonance self-shielding, bound and free atom scattering, and an assortment of nuclear and atomic models, that allow transport calculations to be

accelerated with no lose in accuracy. Indeed it is fair to says that over a given, fixed period of running time TART's results are more accurate than other codes that start from basic principles, and as a consequence they can end up converging extremely slowly, and when they do eventually converge it is usually to the TART answer that you obtain much earlier.

This combination of a modern up-to-date code that runs on any computer, and the best available nuclear models, and nuclear and atomic data, makes today's TART system hard to beat for performance speed and accuracy. The transition from the older **TARTND** code to today's **TART 2002** code did not take place overnight; basically this transition began about seven years ago with the release of **TART95**.

The original **TARTND** code has been used and distributed from Lawrence Livermore National Laboratory for many years. **TART95**, released in July 1995, was the first version of the code designed to be used on virtually any computer. Subsequent release of **TART96, TART97** and **TART2000** were designed to extend the general utility of the code to more areas of application, by concentrating on improving the physics used by the code, particularly with respect to newer neutron data, and more detailed neutron data. **TART2000** further improved the physics, particularly with respect to newer photon data, and more detailed photon data. **TART 2002** is a major step forward in terms of generalizing the nuclear and atomic data used, and greatly expanding the options available to define geometry. TART 2002's new input options, and greatly improved consistency checking are designed to make the code more user friendly and to improve the reliability of results.

TART 2002 completely supersedes all older versions of TART, and it is strongly recommended that users only use the most recent version of TART 2002 and its data files.

For a historical review of the improvements in TART since the release of the first computer independent version, TART95, see the TART2000 report [8c]. Here I only describe where we are today, in terms of currently available features, and I review the new conventions and input options; for complete details of input, see chapter 2 of this report on Input Parameters.

Features of TART 2002

A COMPLETE SYSTEM: The TART system is not merely one single code to run Monte Carlo radiation transport calculations. It is a complete system, that help you check your input before you run calculations (this greatly improves reliability), simplifying running calculations (no complicated system dependent conventions to remember), and help you analyze results, by using interactive graphics, allowing you to overlay your results onto your geometry (this gives you the "big picture", that allows you to see the overall variation of energy deposition, flux…).

Part of this complete system is one, and only one, set of neutron and photon data for use in your applications. I have used my expertise to select what I consider to be the BEST

data for use in applications, and this is what's supplied with TART. TART is designed as an application system, including the BEST code and BEST neutron and photon data that I can provide. There are any number of codes available that can be used to investigate the results based upon using one or another set of nuclear data; TART is not one of these codes. If you want to investigate the effect of using one cross section versus another I suggest you use some other code. Once you have verified that one set of cross sections is better than another, PLEASE, let me know, and I'll consider revising TART's data bases.

COMPUTER INDEPENDENT: TART 2002 today runs on every computer: large central computers, a variety of UNIX workstations (SUN, SGI, HP, DEC Alpha, Meiko, and IBM RISC), Power MACs and IBM PCs using Windows or LINUX. TART is written in such simple and standard FORTRAN that I feel confident to say that not only does TART run on all computers today, it will also run on new computers as they become available; so you can be sure that TART will be available for your use not only today, but also well into the future.

I wasn't too long ago that Monte Carlo radiation transport codes could only be run on multi-million dollar central computers. Today my lap top computer allows me to take TART with me anywhere and to run enormous problems in a small fraction of the time it took on large central computers just a few years ago.

Not only is TART handy as far as its portability, you should also note the economic advantage of today's TART over the old TARTND code that only ran on CRAY computer. For example, my $2,000, 1200 MHz Laptop computer runs TART 2002 over **forty** times as fast as TARTND on a multi-million dollar CRAY-YMP.

The last point concerning computer independence that I'll mention relates to the many users who have asked me, isn't more time consuming and inefficient to make TART so computer independent. He answer is, no it isn't. If anything being able to test TART using many different computer/compiler combinations makes my job easier, and makes TART far more reliable. Different compilers check for different possible errors or inconsistencies, so that I can find and fix as many problems as possible by using as many different computer/complier combinations as possible, making TART more reliable, and ultimately saving myself time and energy.

BEST OPTIONS USED AS DEFAULTS: In the past the users had to be aware of and set all sorts of options and set them in their input files. Now the defaults are what I consider to be the "BEST" set of options, so you need no longer be concerned with them in your input. These options include ALL energy limits for transporting and scoring neutrons and photons; these limits are now automatically set by the code to track and score over the entire energy range of the data in the neutron and photon data files. In addition the important options for resonance self-shielding, thermal scattering, and fluorescence, are initially to have these features turned on.

NEW INPUT OPTIONS: have been added to TART to allow reactor noise analysis calculations (**sentl 54 and 55**), easier simulation of detectors (**sentl 56**), use either total or prompt nu-bar (**sentl 57**), and definition of zone volumes by user input (**volume**).

Options have also been added to extend TART's capabilities for geometry, as well as to simplify and make input more user friendly. These options include: **cubic and quartic (e.g., torus) surfaces,** as well as macro "surfaces" **xyzbox, xcan, xconic**, and macro "volumes", **xsurf**, new **rotation** and **spatial translation**, **surface cloning**, new neutron and photon **sources**, see, **Appendix D: Summary of New Conventions and Options.**

ENDF/B-VI NEUTRON AND PHOTON CROSS SECTIONS: Older versions of TART only used the Livermore ENDL library neutron data, which is primarily designed for use in high energy neutron applications. In contrast the ENDF/B-VI neutron data is designed for general use at all energies [3]. Therefore using this data allows the code to be accurately used in a wider range of applications.

Currently TART 2002 using the ENDF/B-VI, Release 8 neutron data [10], and well as the EPDL97 photon interaction data, which has now been adopted as the ENDF/B-VI standard [6]. Not only is this the most recent release of ENDF/B-VI data, it has also been announced to be the last release of ENDF/B-VI data. Planning is now under way for ENDF/B-VII; this will involve major changes in the ENDF/B system, and will take some time to become a reality.

If you are a fan of the older ENDL data, sorry, but this no longer supported by TART.

TEMPERATURE DEPENDENT NEUTRON DATA: In the past TART has always used nominally room temperature (300 Kelvin) neutron cross sections. We can now prepare additional data files at virtually any temperature to meet programmatic needs [5].

NEUTRON 700 GROUP TREATMENT: for cross sections over the energy range 10^{-5} eV up to 1 GeV. Older versions of the code used a 175 group treatment from 1.309 10^{-3} eV up to 20 MeV, with most of the groups concentrated at higher energy; this limited accurate use of the code to higher energy applications. In contrast the current 700 group treatment is designed to accurately treat the entire neutron energy range, thereby allowing the code to be used for a wider range of applications. The 700 group structure is 50 groups per energy decade, equally spaced in the log of the energy, from10^{-5} eV up to 1 GeV. As yet neutron data is only generally available up to 20 MeV, but as soon as higher energy data becomes available TART is ready to use it.

If you are a fan of the older 175 group treatment, sorry, but this no longer supported by TART.

PHOTON 701 POINT TREATMENT: for cross sections over the energy range 100 eV up to 1 GeV**.** Older versions of the code used a 176 point treatment from 100 eV up to 30 MeV, with most of the points concentrated at higher energy; this limited accurate use of the code to higher energy applications. As with the 700 group neutron treatment, this new treatment of the photon cross sections is designed to accurately treat the entire energy range, allowing the code to be used for a wider range of applications. The 701 points is a fixed set of energy points, 100 points per energy decade, equally spaced in the log of the energy, from10^{0} eV up to 1 GeV. Currently photon data is available up to 1 GeV, so that

all 701 points are currently used by TART.

If you are a fan of the older 176 point treatment, sorry, but this no longer supported by TART.

IMPROVED NEUTRON FISSION TREATMENT: TART includes continuous energy sampling of fission spectra. Earlier versions of TART used equally probable energy bins; equally probable energy bins leads to rather large errors in the low and high energy emission of neutrons. This is a win-win improvement in TART2, where the sampling is now both more accurate and faster. TART also includes the option of use either the average number of neutrons emitted per fission, or to sample from the distribution of the number of neutrons emitted per fission (**sentl 54**). Also included in TART is an option to allow spontaneous fission sources (**sentl 55**). The combination of allowing spontaneous fission sources and sampling from the distribution of the number of neutrons emitted per fission, allows TART to be used to simulate reactor noise analysis problems involving neutron emission correlation effects. You now also have your choice to use either total nu-bar (prompt plus delayed), or prompt (ignoring delayed); this is controlled by **sentl 57**, with the default being total nu-bar. Appendix C illustrates the effect on reactivity of using total versus prompt nu-bar.

NEW UNRESOLVED RESONANCE REGION SELF-SHIELDING: TART includes an extension of the multiband self-shielding method [7] to the unresolved resonance region. See Appendix A for an illustration of the effect that this has on neutron cross sections.

IMPROVED THERMAL SCATTERING TREATMENT: for both bound and free atom scattering. ENDF/B-VI thermal scattering law data is only available for a few materials [12]. TART now includes six of these in its neutron data library,

ZA = 1801 H bound in H2O
1901 H bound in CH2
1902 D bound in D2O
4809 Be bound in Be metal
4909 Be bound in BeO
6912 C bound in graphite
8916 O bound in BeO

To use these in your applications you need merely specify the above ZA in place of the normal ZA when defining your materials in TART input. For example, for free H in H2O the input is two parts ZA = 1001 (H) and one part ZA = 8016 (O); for bound H in H2O you need merely specify two parts ZA = 1801 (H bound in H2o) and one part ZA = 8016 (O). The except being that for bound BeO you must specify both Z = 4909 (Be bound in BeO) and ZA = 8916 (O bound in BeO) – above list includes seven items, but really only includes data for six materials, since for BeO both Be and O MUST be defined as bound.

Since thermal scattering law data is available only for a few materials, free atom scattering is still important, since it is used for all other materials. The major advantages

of the new free atom thermal scattering treatment include: improved accuracy of sampling, and greatly improved speed of execution [4].

PHOTON SCATTERING TREATMENT: TART includes an improved treatment of photon coherent and incoherent scattering. The current treatment has the advantage that it is both more accurate, and faster to use, than the older treatment.

IMPROVED GEOMETRIC EXPERT SYSTEM: TART has always included an expert system that learns what your geometry looks like as your problem runs. This allows tracking through your geometry to accelerate during any given TART run. TART allows you to carry this learned information forward to successive TART runs. The information defining the connectivity of your geometry is automatically output to be file named **GEOLINKS**, and if you run a problem using the same geometry in the future, this information is read by TART, to allow it to accumulate experience and accelerate even further with each successive TART run.

IMPROVED NO UPPER OR LOWER LIMITS: on anything you can define by input. Unlike earlier versions of TART, that had a maximum allowed number of zones, surfaces, etc., today's TART has no limits at all. For example, earlier versions of the code were limited to a maximum of 1,000 spatial zones. Since then the spatial detail used in TART problems has increased enormously. The largest TART problem that I know of involved 27 million (27,000,000) spatial zones. Of course TART can still accommodate even the simplest problem, such as a one zone spherical ball. In all cases from smallest to largest TART98 automatically sizes itself to accommodate each individual problem run.

LONG RUN RANDOM NUMBER GENERATOR: The current random number generator includes over 2,500 different random number sequences, each sequence a trillion (10^{12}) random numbers displaced from the preceding sequence. With a modern computer we can generate a trillion random numbers in about one day, if that's all a code is doing. With TART each random number sequence should take about 10 to 20 days to use a complete trillion number sequence. Therefore the currently available 2,500 sequences should keep you busy for years; and if you need more, just ask for them.

MULTIPROCESSING: The TART utility codes **MULTIPRO** and **TARTSUM** are now routinely used to perform multiprocessing. This approach to multiprocessing is so simple, straightforward and general that virtually all TART users can use it. With this approach if you have a computer with thousands of processors you can use **MULTIPRO** to create everything that you need to use as many processors as you want and then average the results together using **TARTSUM**. Even if you don't have a computer with many processors, but you do have access to a number of computers, you can use all available computers to run problems (they don't even have to be the same type of computer), and again use **TARTSUM** to average all of the results together. This approach is completely computer independent, and in the example case of using 250 processors, you can compress 250 days of work into a single day (more than a year of working days into one day). With the new random number generator, using different random number sequences for each run, you can make over 2,500 statistically independent runs and combine the results.

Unlike the approaches used by other codes that are tied to specific computers, and have overhead due to communications between processors, with this approach you can use as many processors as are available, all at the same time, or at different time, and there is no overhead due to communications between processors, since each run is independent and contributes to the final results. You can do this simply by running the same problem with different random number sequences, either using multiprocessing, or any number of single processor computers that you have access to, or a single processor repeatedly, if you just want to run more histories to improve your results. This gives you the ultimate flexibility as far as how to best use the computer resources available to you.

IMPROVED INPUT CHECKING: to catch more input errors before the calculation begins. As described below, this checking is now incorporated in both TART and TARTCHEK. TART continues the TART traditions to support all older TART input. For example, if you have a twenty year old TART input problem, you will still be able to use it with TART. However, TART and TARTCHEK are now much more clever at finding errors in TART input, with the result that you may find that TART input decks that ran earlier, will now cause TART to stop, with detailed ERROR messages asking you to correct your input before proceeding.

MODULAR CODING: TART is also very **modular**, so that portions of the code can be used in other codes. For example, TART and TARTCHEK use exactly the same input and geometry package; this assures that when you use TARTCHEK to check your geometry, when you run TART it will interpret your geometry in exactly the same manner.

INTERNAL CONSISTENCY CHECKING: No code is perfect, and for any complicated code, such as TART, that has many possible paths through it, it is virtually impossible to manually check all possible paths. With TART's internal consistency checking, it does its own checking every time it is run. For example, every single array in the code is checked for misuse, in an attempt to find as many errors as possible. This procedure has already led to accelerated code development and improvements, and will continue to do so in the future.

INTERNAL CONSISTENCY CHECKING: No code is perfect, and for any complicated code, such as TART, that has many possible paths through it, it is virtually impossible to manually check all possible paths. The code now includes internal consistency checking; it does its own checking every time it ran. For example, every single array in the code is checked for misuse, in an attempt to find as many errors as possible. You wouldn't believe how effective this internal checking has been over the last few years at finding and allowing us to eliminate potential problems, resulting in a much more reliable code. This internal consistency checking has been so effective that since the release of TART98 not one single internal consistency problem has occurred. However, I have left the internal consistency checking in place within the code, so that any changes or new additions to the code are automatically verified.

FULL OPTIMIZATION: One general improvement worth noting, is that based on

communications with a variety of FORTRAN compiler designers, TART's has been re-designed to allow it to be compiled at the **highest level of optimization** on most computers, which can greatly reduce running time, without sacrificing accuracy.

CONTINUATION OF ANY INPUT LINE: Any input line can now be continued onto any number of continuation lines. With earlier versions of TART some input, particularly complicated sources, could not be continued from one line to another, which made input preparation difficult. You will find that being able to continue any input line, it is much easier to prepare input. Some of the following new options, such as cloning, rotation and spatial translation, were recommended by TART users, and are also designed to simplify preparation of TART input. If you have ideas to even further simplify input preparation, I'd love to hear them.

GENERAL IMPROVEMENTS: TART is based on the older TARTND code, but required massive changes to the code to make it the modern, computer independent code that it is today. As such there were bound to be some growing pains with this essentially new code. Over the last few years, feedback from the many code users has led to general improvements in the code, both in terms of locating and correcting problem areas, as well as in adding and improving code options to meet the needs of users. Examples of overall corrections and improvements include,

1) Corrected Be-9 (n,2n) neutron emission spectra, which were incorrect.
2) Corrected Z cone Reflection.
3) Corrected addxyz input.
4) Added zone volume user input option.
5) Added the option with tally types 11 and 12, to tally in zone neutron absorption, rather than every time a neutron enters a zone.
6) Allowed type 11 and 12 for empty (no material assigned) zones.
7) Added improved reflected calculations.

EXPERIENCE: least, the other important change is EXPERIENCE!!! Again, I cannot stress how important this is for any code. In the case of TART each successive version of the code includes the operating experience of the many people who are now using the code. With each passing version of TART reliability and accuracy are improved, mostly based on feedback from users - such as you – so keep that feedback coming.

Running Time

The below table presents results obtained using a collection of 68 TART benchmark problems. All 68 problems were run on each computer. This table summarizes timing results for the older TARTND code that only runs on CRAY computers, as well as TART 2002 and TART95 on a variety of computers.

Code	Computer	Running Time (Seconds)	Ratio to TARTNP CRAY-YMP
TARTNP	CRAY-YMP	5396	1.0
TARTNP	CRAY-J90	7727	1.43
TART 2002	Athlon XP1800/1520	89	0.0165
TART 2002	IBM-PC Pentium IV/2000	132	0.025
TART 2002	IBM-PC Lap Top III/1200	133	0.025
TART 2002	IBM-PC Pentium III/1000	170	0.031
TART 2002	IBM-PC Pentium III/500	500	0.09
TART 2002	DEC-Alpha Model 5/625	516	0.10
TART 2002	IBM-PC Pentium II/400	579	0.11
TART 2002	PowerMAC/LapTop/500	683	0.126
TART 2002	IBM-PC Pentium II/333	697	0.13
TART 2002	DEC-Alpha Model 5/300	712	0.13
TART 2002	IBM-PC Pentium II/266	855	0.16
TART 2002	IBM-PC Pentium Pro/200	1185	0.22
TART 2002	IBM-PC Lap Top/233	1301	0.24
TART 2002	Power-MAC 7500/275	1350	0.25
TART 2002	iMAC	1664	0.31
TART 2002	HP-735/125	1834	0.34
TART 2002	SUN E3000/166	2107	0.39
TART 2002	IBM-PC LapTop/133	2990	0.58
TART 2002	CRAY-YMP	4262	0.79
TART 2002	IBM-RISC RS-6000	5739	1.06
TART 2002	CRAY-J90	6095	1.13
TART 2002	Meiko CS-2/66	6225	1.15
TART 2002	SUN Sparc-20	6315	1.17
TART 2002	Power-MAC 7500/100	6446	1.21
TART 2002	SGI R4000/100	6953	1.29
TART95	CRAY-YMP	4912	0.91
TART95	HP-350	4322	0.80
TART95	DEC-Alpha	6130	1.14
TART95	SUN	9673	1.79
TART95	Meiko	9993	1.85
TART95	SGI	10157	1.88
TART95	IBM-RSIC	14838	2.75
TART95	IBM-PC 486DX2/66	18437	3.41

When we compare the three codes all run on the same CRAY-YMP, we find that compared to the older TARTND code, TART95 was about 9 % faster, and TART 2002 is about 21 % faster. So that not only has been extended for more general uses, these extensions were accomplished with no lose in running time efficiency, i.e., TART 2002 is actually faster than TART95.

You should also note the advantage of TART95 and TART 2002 over the older TARTND in terms of their ability to be used on virtually any computer. For example, a $2,000 Pentium III, 1200 MHz Laptop computer runs TART 2002 over **forty** times as fast as TARTND on a CRAY-YMP, and the Athlon 1500 MHz computer runs TART 2002 over **sixty** times faster.

Why is Monte Carlo Used so much Today?

The last point to note from these comparisons is how far we have come in terms of available inexpensive computer power in the few years between the release of TART95 and TART 2002. When TART95 was released the fastest IBM-PC then available took 18,437 seconds to run this collection of 68 problems. Even then we could foresee the potential of an inexpensive computer being able to run these problems in only about 3.4 as much time as it took on a CRAY-YMP. But I don't think anyone could foresee that just a few years later we would have available IBM-PCs that can run this collection of problems in only 89 seconds. Compared to the PCs of only a few years ago, not only does today's PC run these problems over **207** times faster, but also it does it at about half the cost.

Think about what a difference in running time of a factor of **207** means. A major expense of any scientific project is your salary, so time is money and it can be expensive - or inexpensive, depending on how you spend it. Consider that only a few years ago if it took an entire 9 to 5, 8 hour working day (480 minutes), to run a TART problem, on an IBM-PC today it would take about **2.3** minutes to run the same problem, i.e., a factor of **207** faster. It should be noted, that this tremendous increase in available inexpensive computer power is one of the reasons that the use of Monte Carlo has expanded so much in recent years. Problems that we thought too time consuming to be practical just a few years ago, have now become routine.

Why is TART so FAST?

Some users make the mistake of assuming that since TART is so much faster than other codes that perform the same types of calculations, the results based on other codes must be better than those based on TART. When you use TART you will find that its results are just as accurate as those of other codes, and often more accurate. So why is TART so fast?

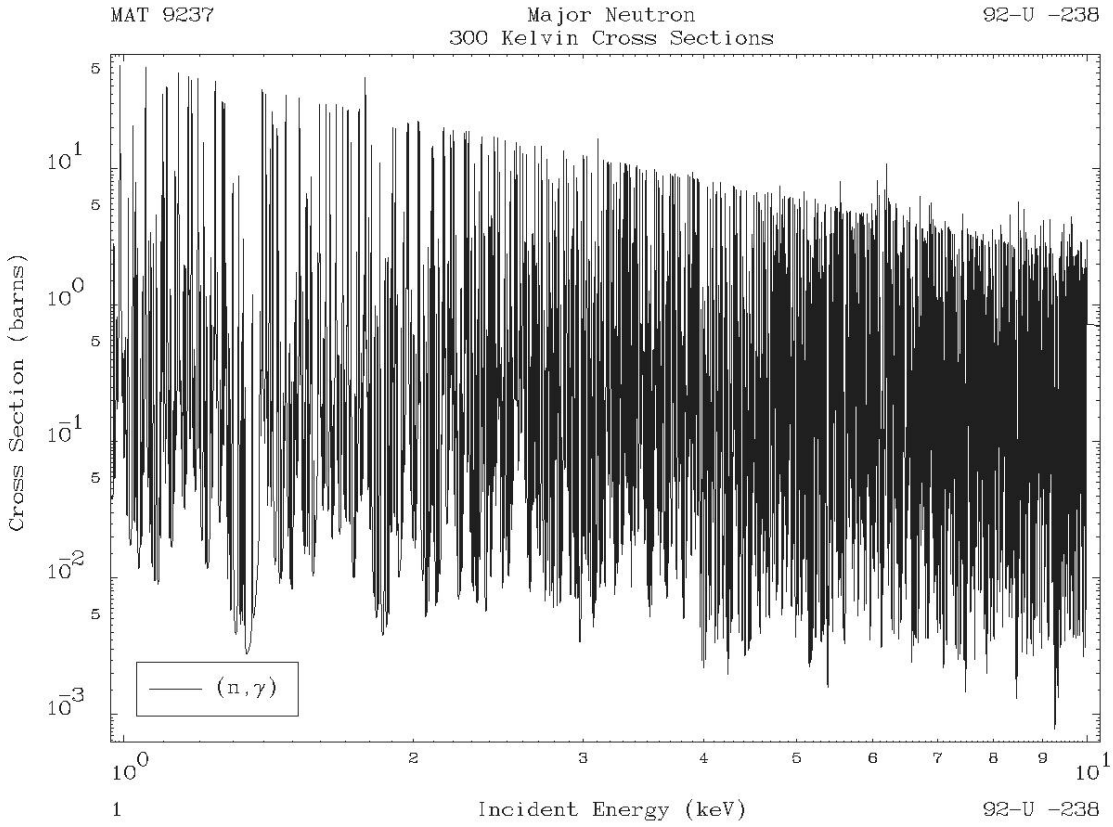
There isn't any big secret to TART's speed: TART includes the three most important things necessary for generally efficient and accurate programming:

EXPERIENCE! EXPERIENCE! EXPERIENCE!

It is as simple as that. TART is based on almost 40 years of continuous use and improvement. During this time roughly 90 work years of physicist/programmer time, and hundreds of work years of user experience, were incorporated into the code that we have today. To illustrate why TART is so much faster and still as accurate as other codes, I'll mention just a few points.

First is the use of multi-group data, including the multi-band method to account for self-shielding [1, 7], as used by TART, compared to continuous energy cross sections used to other codes. Results using continuous energy cross sections have to be better, right? Not always! This is only true if you run a calculation for extremely long times so that you

accurately sample ALL of the continuous energy cross sections. This is almost never done, and I know of no code that explicitly includes an estimate of the uncertainty in its results based on the enormous variation in continuous energy cross sections. In comparison, TART's approach is designed for the real world, and incorporates not only the best nuclear and atomic data, but also the best nuclear and atomic engineering.



For example, if we look at the U-238 cross sections we see capture cross sections even over a fairly small energy range of 1 to 10 keV, that vary by roughly four orders of magnitude, and we can see that it is composed of very narrow resonances with relatively large energy intervals between resonances, i.e., the ratio of resonance spacing to width is about 100 to 1. This data is VERY DIFFICULT to sample on a continuous energy basis. Indeed if you try it you will find that in order to obtain even a fairly accurate estimate of the average cross sections and distance to collision you would have to sample billions of histories. I don't know of any code that uses continuous energy cross sections that actually does this. They simply supply you with the "best" possible cross sections and assume that this will solve your problems.

TART takes it a step further: not only does TART use the "best" cross sections, but also uses the "best" nuclear engineering. Again, consider the U-238 cross sections. Anyone who has taken a course in reactor physics knows that in this case the neutron flux will self-shield and we know the form of the self-shielding. Therefore we do not need all of the nitty-gritty details of each and every narrow capture resonance in order to perform an

accurate transport calculation. Think about it: people have been successfully designing nuclear reactors for over 50 years, and yet only fairly recently have detailed cross sections become available. So how did people design their reactors? They did what TART now does: combine the "best" currently available nuclear data with the "best" nuclear reactor theory. In the case of TART the use of the multi-band method to account for resonance self-shielding [1, 7] allows it to use multi-group, rather than continuous energy cross sections, resulting in rapid convergence of calculations, compared to code that use continuous energy cross sections and take forever to converge. Most important for users to understand is that this is done with virtually no lose in accuracy in the TART calculations, indeed it is fair to say that since for reasonable running times the TART results converge and those of other codes do not, from the pragmatic viewpoint of obtaining accurate answers in a reasonable amount of time, the TART results are better.

I should also mention the unresolved resonance region, where by definition we do not know the cross sections on a continuous energy basis, but it can be accurately treated by the multi-band method used by TART. Ask yourself: what do codes that claim to use continuous energy cross sections do in the unresolved resonance region?

A second example of why TART is so fast is its treatment of geometry. Compared to other codes TART uses a very strict geometry, which places an additional burden on the user in terms of input preparation. But the pay off is that the input is easier to check and correct (using TARTCHEK) to improve reliability, and when the code starts to run it FLIES!!!

For example, TART insists that the users define every space point to be within a spatial zone. Other codes do not insist on this, so why does TART? The first reason is that without insisting on this it is not possible to check the input parameters for errors; checking is now simple and straightforward using TARTCHEK, and greatly improves the reliability of the input. Next, when TART runs it greatly accelerates tracking. How can a few holes in the geometry make such a big difference? Consider a simple problem involving 1000 spatial zones with each zone bounded by 6 surfaces. When a particle enters a spatial region that is not defined in the problem, i.e., is a "hole", the code has to track (ray trace) to the nearest bounding surface to determine what zone it will next enter. In this example it has to ray trace to the 6 bounding surfaces of each of the 1,000 spatial zones, to determine which of these surfaces is closest to the particle in its direction of travel, i.e., it has to ray trace to 6,000 surfaces. In contrast, with TART where a particle is always within a defined zone, in this example, we are inside one of the zones and we have to track (ray trace) to the nearest boundary of the zone. This only involving tracking to each of the bounding surfaces of this one zone, i.e., ray trace to 6, rather than 6,000 surfaces. No wonder TART geometry is so much faster to track through. How much of an effect does this really make? TART and TARTCHEK use exactly the same geometry package. In the original method used by TARTCHEK to display 3-D objects, TARTCHEK used a general ray tracing technique that did not take advantage of TART geometry. When TARTCHEK was updated to take advantage of TART geometry the ray tracing to display 3-D objects ran up to 200 TIMES FASTER - not 200 % - 200 TIMES!!! Pictures that took hours or all night to produce could suddenly be done in minutes or seconds. The difference was dramatic. You can see for yourself; use TARTCHEK to display 3-D views of your geometry and you will be amazed at how fast

it can do it - and remember in doing it, it is using EXACTLY the same routines that TART uses to track through 3-D geometry. No wonder TART is so fast.

These are but a few examples of why TART runs so much faster than other codes, with essentially no lose in accuracy. Try it for yourself and see what you think.

As related to reliability, I'll also mention in passing the danger of using the default of other codes that assume that whatever volume you have not explicitly defined is vacuum that the code can freely transport through. My experience has been that when a problem has an undefined volume in it, well over 90 % of the time it is because it is an error. Other codes sweep this under the rug and make it appear that nothing is wrong, usually resulting in the wrong answer. In contrast TARTCHEK and TART 2002 will quickly find these volumes and ask you to explicitly define them. This approach greatly improves the reliability of the TART input.

What Code should you be using?

TART 2002 completely supersedes all older versions of TART, and it is strongly recommended that users only use the most recent version of TART 2002 and its data files. How do you know if you have the most recent version of the code and its data files? As soon as the code starts to run it identifies the version you are running and the dates of its data files. Below is the beginning of the code output report. Note, the code version: **TART02-4, Dec. 2002**, and the date of the four data files is **03/03/03**. Note, also the newer **616 groups** for the neutron data and **701 points** for the photon data. If you are using an older version of the code or its data files, it is strongly recommended that you obtain the most up-to-date code and data; see, the below section on **Availability**.

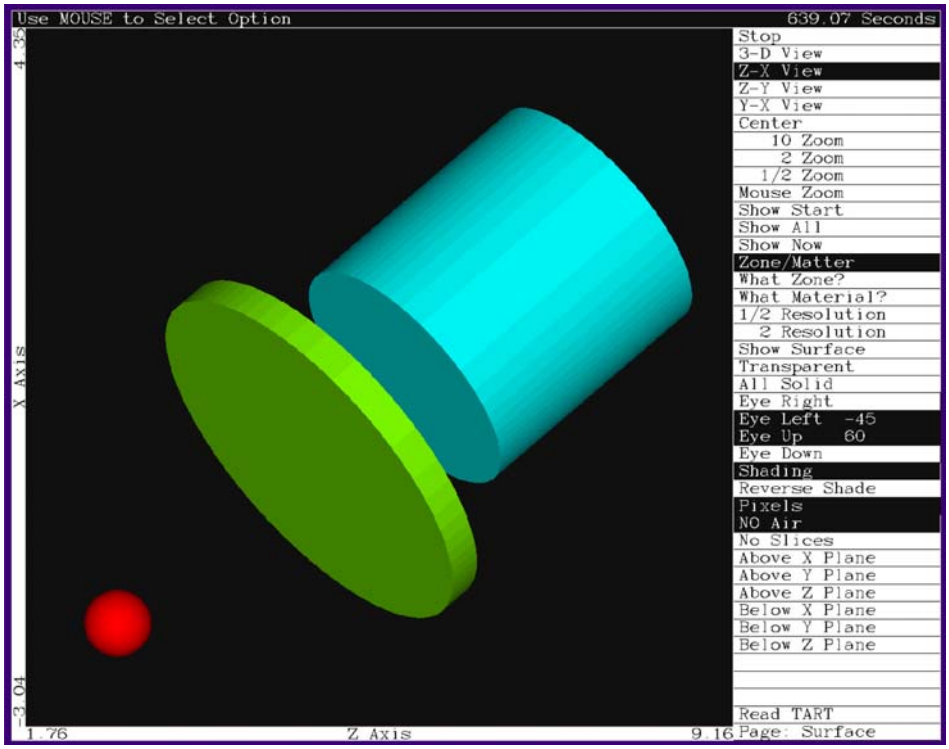
```
TART 2002 - Neutron-Photon Monte Carlo Transport (TART02-4, Dec. 2002)
================================================================================
 I/O Files Opened for Entire Run
================================================================================
 Definition                                  Filename      Unit  Date
================================================================================
 TART Input Parameters.....................fastcrit         2
 TART Output Listing.......................TART.OUT         3
 TART Input Scratch File...................fastcrit.TMP    33
 Geometry Connectivity File................GEOLINKS         11
 Neutron Interaction Data File.............TARTND            7  03/03/03
 Photon Interaction Data File..............GAMDAT            8  03/03/03
 Neutron Induced Photon Production File...TARTPPD           9  03/03/03
 Multi-Band Parameter File................NEWCROSS         10  03/03/03


 Neutron Interaction Data.  616 Groups 1.0000D-11 to 2.0000D+01 MeV
 Photon Interaction Data..  701 Points 1.0000D-04 to 1.0000D+03 MeV
```

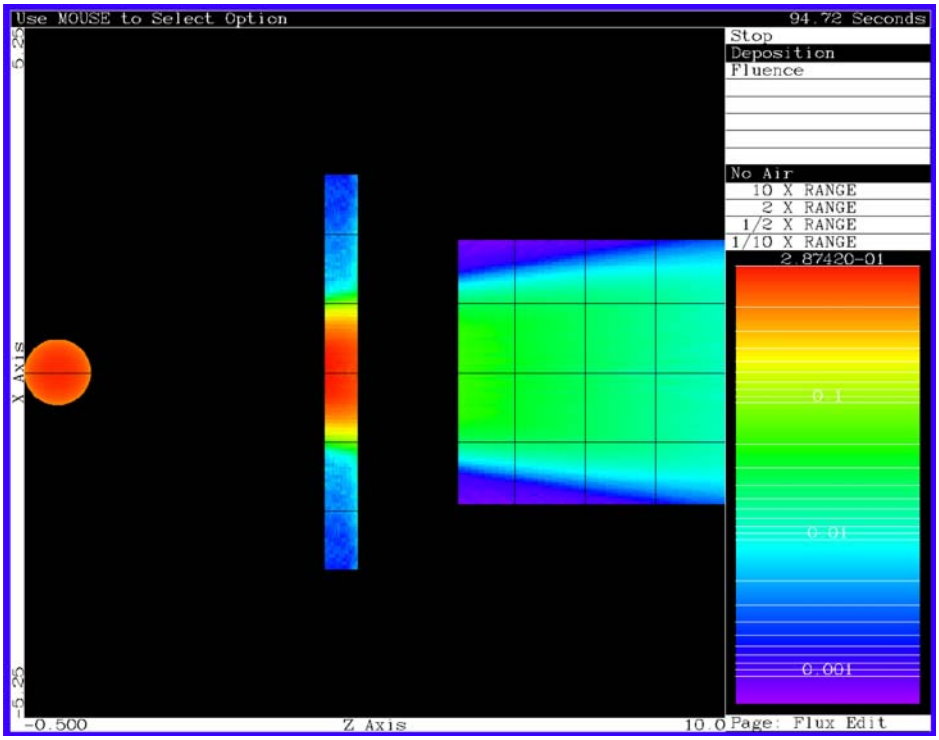
Utility Codes

In addition to the TART code you should also be aware of the utility codes distributed with TART; of particular note are **TARTCHEK** and **TARTAID**. One of the most difficult tasks that you will face in using any 3-D combinatorial Monte Carlo code is to correctly define input parameters for the code, particularly to correctly define geometry. This is what **TARTCHEK** is designed to help you with. It is an interactive graphics code that will allow you to view and check your input parameters before you run **TART**. Even we so called "experts" on **TART** find that using **TARTCHEK** can greatly reduce the amount of time that we have to spend on input preparation, and even what is more important, greatly improve the reliability of our input parameters. In addition the TART 2002 CD system includes **TARTAID**, which will allow you to interactively create TART input decks from scratch.

By now most, if not all, TART users are familiar with **TARTCHEK**'s ability to show you your geometry in 2-D, and to quickly test for error in your TART input parameters. But many users are still not familiar two additional capabilities of TARTCHEK as shown below: first its ability to show you geometry in 3-D, and second its ability to overall

results of a TART calculation on your geometry. Particularly the latter is an extremely powerful tool to help you "see" the big picture, as far as global variations of energy deposition and flux. Instead of spending days or weeks wading your way through a thick output listing trying to understand the results, using **TARTCHEK** a few minutes after you finish a TART calculation you can "see" the results overlaid on your geometry. Not only will this save you time, it can improve your overall understanding of the results, by showing you the "big picture" of how flux, deposition. etc., in each zone is related to that in all other zones. This is something that is very difficult to "see" regardless of how long you stare at an output listing. If you are not using **TARTCHEK** you are only making your job more difficult, and you don't know what you are missing.





TARTAID is another code you should be aware of. In addition to **TARTCHEK**, which can be used to check existing TART input, and display TART results, the TART 2002 CD system also includes **TARTAID**. This code is designed to help you create TART input from scratch. It is particularly helping to define very detailed geometry, involving many spatial zones. For example to create a TART input deck involving 10,000 or even 100,000 spatial zones, takes only minutes using **TARTAID**.

You should also be away of the utility codes **MULTIPRO** and **TARTSUM**, which will allow you to easily run many TART problems simultaneously, and then add together results from any number of TART problems, and produce a combined output file in **EXACTLY** the same format as any other single TART problem output file. Our computers are getting faster and faster, but we are running into the speed of light problem, where we can only get so much work done using a single processor. TART's approach to multiprocessing allows us to avoid this limit, in the sense that we can now compress the work that used to take many days, into a single day. This is true on either multiprocessing computers or a group of single processors computers. Just run your problems on ANY computer(s), using as many processors as you have access to, and **TARTSUM** will combine the results for you. Note, since the combined output file produced by **TARTSUM** is in **EXACTLY** the same format as any other single TART problem output file, if you are one of the many TART users who have utility codes to further process TART output results - not to worry - your utility codes will work on the combined file, exactly the same way they work on the results of a single TART run.

Documentation

Although TART 2002, supersedes all earlier versions of TART, the most complete

documentation for TART is still,

TART95: A Coupled Neutron-Photon Monte Carlo Transport Code, Lawrence Livermore National Laboratory, UCRL-MA-121319, July 4, 1995, by D. E. Cullen, A.L. Edwards and E.F. Plechaty

This document, as well as all other TART documentation, is now available on-line at the TART website,

htp://www.llnl.gov/cullen1

Availability

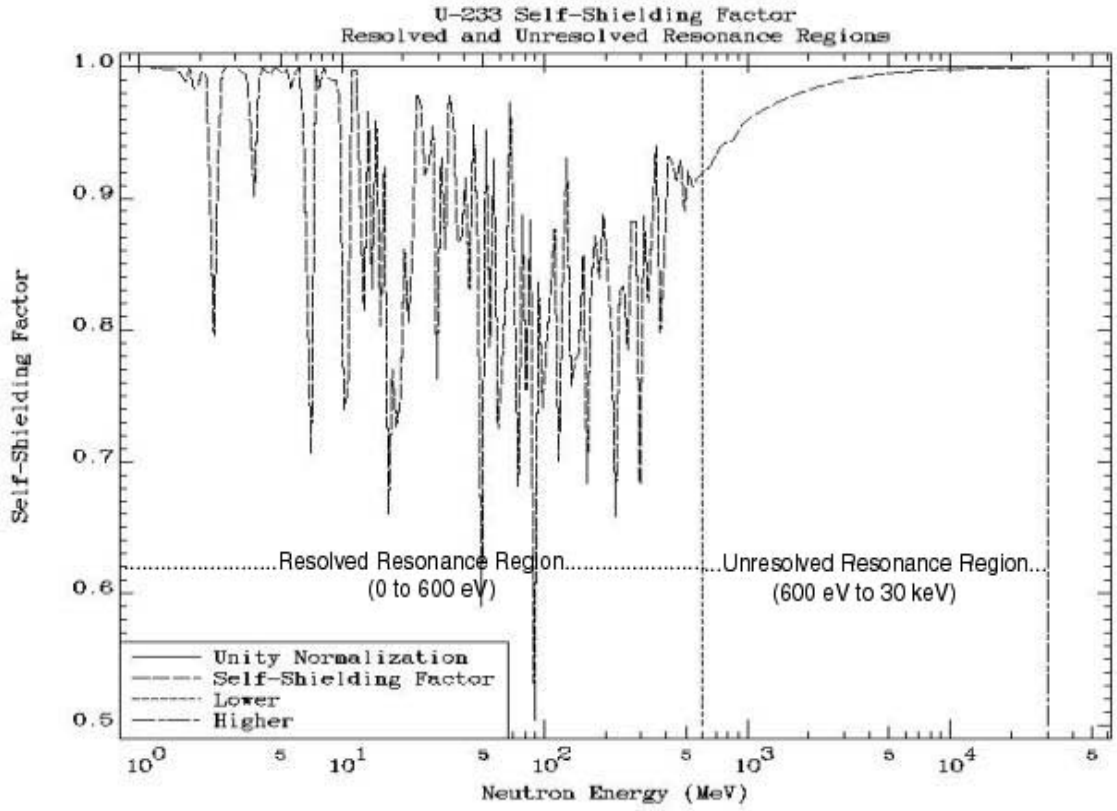
At Livermore, for copies of the system, contact Red Cullen, x-3-7359. Outside of Livermore, contact your local computer code center - within the United States, the Radiation Safety Information Computational Center (RSICC), Oak Ridge National Laboratory (e. mail: jib@ornl.gov), outside of the United States, the OECD Nuclear Energy Agency/Data Bank (NEA/DB), Paris, France (e. mail: sartori@nea.fr).

Code Installation

The code is distributed with detailed instructions concerning installation and testing of the code. These instructions are periodically updated for distribution with the code, to insure that the instructions are as up-to-date as possible, and exactly correspond to the version of the code that you will be implementing and using. As such, installation instructions will not be included here.

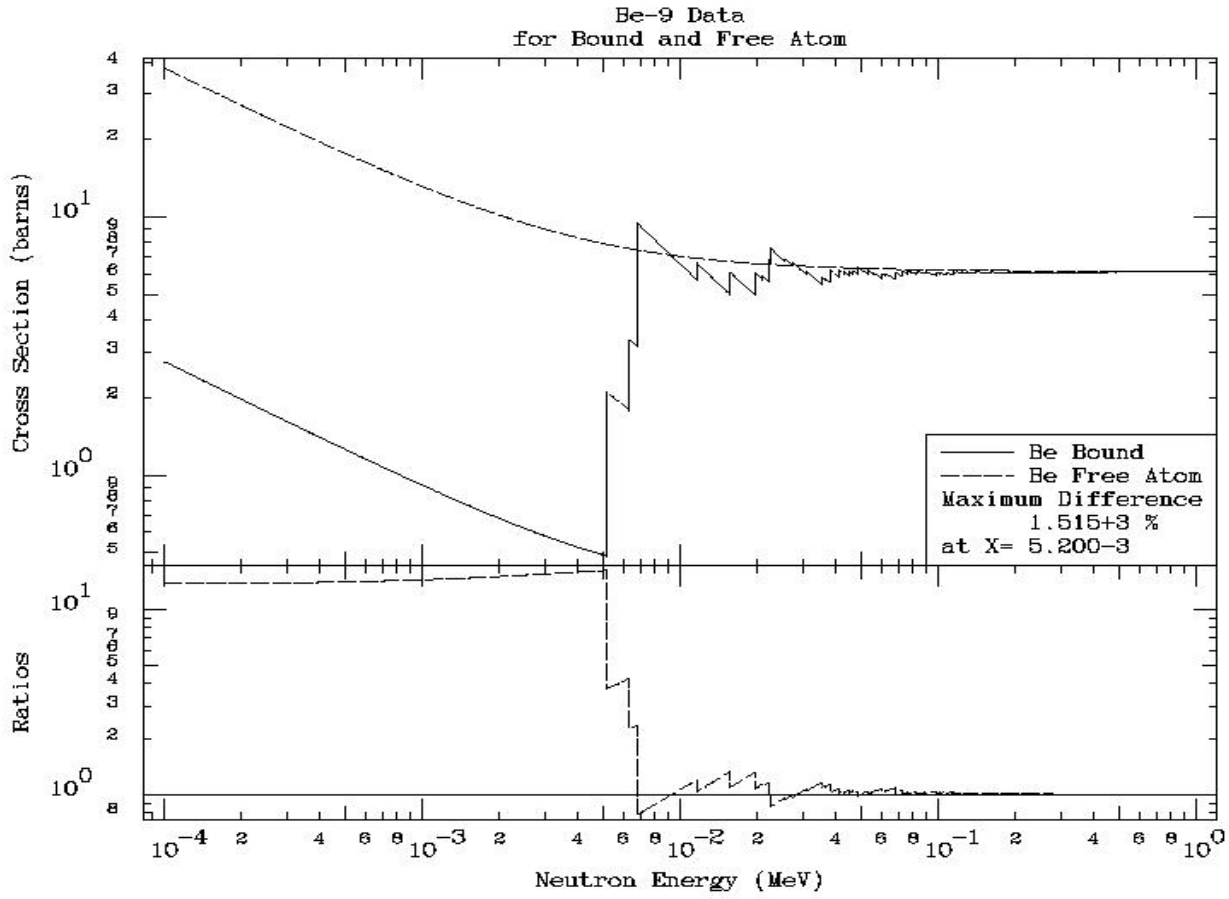
Appendix A: Illustration of Resonance Region Self-Shielding

The below figure illustrates the effect of resonance self-shielding on the 700 group U-233 cross sections used by TART. 700 groups may seem like a lot and you might think that there would be little or no resonance self-shielding. If this were the case the self-shielding factors, which are defined as the ratio of self-shielded to unshielded cross section, would be unity in all groups. The below figure illustrates that this is not the case, because the 700 groups are not nearly enough to accurately represent the energy dependent shape of the many narrow resonances in U-233. In the case of U-233 the resolved resonance region extends up to 600 eV, and the unresolved resonance region extends from 600 eV up to 30 keV (indicated by the two vertical lines on the below figure). Note, that the current TART data includes self-shielding in both the resolved and unresolved resonance regions.



Appendix B: Comparison of Bound and Free Atom Be-9 Data

The below figure illustrates the effect of molecular binding on the total cross section of Be-9. At higher energies, above the eV energy range, the effects of binding are negligible, and the cross sections are identical. At lower energies note the discontinuities in the bound cross at Bragg edges, and below about 5 milli-eV the bounding cross section deceases by over an order of magnitude.



Appendix C: The Effect on Reactivity using Total versus Prompt Nu-bar

The below table illustrates the difference in reactivity (K-effective) of the 68 criticality problems distributed with TART using either total (prompt and delayed) or only prompt number of neutrons per fission (nu-bar). Note that the difference depends on the type of fissile material used, where the delayed fraction is small for Pu-239, intermediate for U-233, and larger for U-235, which can be seen by the magnitude of the differences in K-effective for these three different materials in a variety of systems.

Problem	Fuel	Reflector		Total	Prompt	Difference
c10100	pu-a	be	5.222	1.000290	0.997982	-0.231 %
c20100	pu-a	be	8.170	1.002780	1.000700	-0.208 %
c30100	pu-a	be	13.000	1.004140	1.002190	-0.195 %
c40100	pu-d			0.997953	0.995940	-0.202 %
c50100	pu-d	be	3.690	0.998140	0.996117	-0.203 %
c60100	pu-d	be	5.250	0.999445	0.997735	-0.171 %
c70100	pu-d	c	3.830	0.998298	0.996177	-0.213 %
c80100	pu-d	ti	8.000	0.986907	0.984861	-0.208 %
c90100	pu-d	w	4.700	0.992985	0.991153	-0.185 %
c10010	pu-d	u-235	0.660	0.999219	0.996183	-0.305 %
c11010	pu-d	u-238	1.930	0.992996	0.990539	-0.248 %
c12010	pu-d	u-238	6.740	0.997434	0.994696	-0.275 %
c13010	pu-d	u	4.130	1.000410	0.997680	-0.274 %
c14010	pu-d	u	19.600	0.998271	0.994706	-0.358 %
c10100	u-233			0.996675	0.993851	-0.284 %
c20100	u-233	be	2.050	1.000000	0.996944	-0.307 %
c30100	u-233	be	4.200	1.002300	0.999646	-0.265 %
c40100	u-233	w	2.440	0.998245	0.995478	-0.278 %
c50100	u-233	w	5.790	0.997022	0.994353	-0.268 %
c60100	u-233	u-235	1.210	1.001660	0.998594	-0.307 %
c70100	u-233	u-235	1.980	1.005640	1.001760	-0.387 %
c80100	u-233	u-235	4.820	1.009170	1.003760	-0.539 %
c90100	u-233	u	2.300	1.002160	0.998909	-0.325 %
c10010	u-233	u	5.310	1.004550	1.001070	-0.348 %
c11010	u-233	u	19.910	1.002520	0.998268	-0.426 %
c001	u-235	be	1.27	0.991333	0.984829	-0.660 %
c002	u-235	be	2.54	0.993284	0.986833	-0.654 %
c003	u-235	c	1.27	1.002000	0.995486	-0.654 %
c004	u-235	c	2.54	1.003290	0.996490	-0.682 %
c005	u-235	mg	1.27	0.991813	0.985385	-0.652 %
c006	u-235	mg	2.54	0.994947	0.988433	-0.659 %
c007	u-235	al	1.27	0.989334	0.982866	-0.658 %
c008	u-235	al	2.54	0.988649	0.982307	-0.646 %
c009	u-235	ti	1.27	0.994135	0.987767	-0.645 %
c010	u-235	ti	2.54	0.996451	0.989906	-0.661 %
c011	u-235	fe	1.27	0.998081	0.991672	-0.646 %
c012	u-235	fe	2.54	0.990955	0.984332	-0.673 %
c013	u-235	ni	1.27	0.990904	0.984588	-0.641 %
c014	u-235	ni	2.54	0.994334	0.987967	-0.644 %
c015	u-235	cu	1.27	0.991635	0.985316	-0.641 %
c016	u-235	cu	2.54	0.998562	0.992127	-0.649 %
c017	u-235	mo	1.27	1.003360	0.996582	-0.680 %
c018	u-235	mo	2.54	1.013120	1.006210	-0.687 %
c019	u-235	mo-allo		1.003810	0.997282	-0.655 %
c020	u-235	w	1.27	0.987416	0.980685	-0.686 %
c021	u-235	w	2.54	0.987675	0.981390	-0.640 %
c10100	u-235			0.998285	0.992167	-0.617 %
c20100	u-235			1.004730	0.997933	-0.681 %
c30100	u-235			1.001100	0.994867	-0.627 %
c40100	u-235	be	2.222	0.998019	0.991442	-0.663 %
c50100	u-235	be	3.260	0.999108	0.992707	-0.645 %
c60100	u-235	be	4.710	1.002990	0.996677	-0.633 %
c70100	u-235	be	5.440	1.000370	0.993858	-0.655 %
c80100	u-235	be	9.270	1.001940	0.995947	-0.602 %
c90100	u-235	be	11.790	1.001540	0.995308	-0.626 %
c10010	u-235	c	10.160	0.997750	0.991028	-0.678 %
c11010	u-235	c	15.240	0.993739	0.987280	-0.654 %
c12010	u-235	ni	4.940	0.997086	0.990638	-0.651 %

c13010	u-235	cu	5.030	1.003730	0.997110	-0.664 %
c14010	u-235	cu	10.560	1.009160	1.002330	-0.681 %
c15010	u-235	w	5.080	1.000260	0.993600	-0.670 %
c16010	u-235	w	10.160	1.001170	0.994547	-0.666 %
c17010	u-235	pb	8.990	1.020890	1.014200	-0.660 %
c18010	u-235	pb	17.220	1.020590	1.013680	-0.682 %
c19010	u-235	u	1.760	1.001340	0.994603	-0.677 %
c20010	u-235	u	4.470	1.005410	0.998442	-0.698 %
c21010	u-235	u	9.960	1.002660	0.995373	-0.732 %
c22010	u-235	u	18.010	0.998976	0.991576	-0.746 %
			Averages	**0.999340**	**0.994163**	**-0.521 %**

Appendix D: Summary of New Conventions and Options

To help explain and illustrate the use of the new options the TART 2002 CD distribution includes example input decks. I encourage you to use TARTCHEK to look at these examples - particularly using 3-D views, so you can see them better.

See chapter 2, on Input Parameters for details. Here I only briefly list the new input options and conventions.

Cubic surface

xcubic nb x0 y0 z0 d c b a
ycubic nb x0 y0 z0 d c b a
zcubic nb x0 y0 z0 d c b a

Torus

xtorus nb x0 y0 z0 a b c
ytorus nb x0 y0 z0 a b c
ztorus nb x0 y0 z0 a b c

Macro surfaces: boxes, cylinders and cones with finite limits

xyzbox nb dx dy dz [x0 y0 z0]
xyzbox2 nb x1 x2 y1 y2 z1 z2
xcan nb rad dx [x0 y0 z0]
ycan nb rad dy [y0 x0 z0]
zcan nb rad dz [z0 x0 y0]
xconic nb rad1 x1 rad2 x2 [y0 z0]
yconic nb rad1 y1 rad2 y2 [x0 z0]
zconic nb rad1 z1 rad2 z2 [x0 y0]

Macro volumes: simplified input for complicated surfaces

```
xsurf   nz   ns    x1   ri1   ro1 &
                   x2   ri2   ro2 &
                   x3   ri3   ro3  &
                        (continued for more points)

ysurf   nz   ns    y1   ri1   ro1 &
                   y2   ri2   ro2 &
                   y3   ri3   ro3  &
                        (continued for more points)

zsurf   nz   ns    z1   ri1   ro1 &
                   z2   ri2   ro2 &
                   z3   ri3   ro3  &
```

(continued for more points)

Rotation about the X, Y or Z axis

xrotate ang is1 thru is2
xrotate ang is1 is2 is3.....
yrotate ang is1 thru is2
yrotate ang is1 is2 is3.....
zrotate ang is1 thru is2
zrotate ang is1 is2 is3.....

Translation of Spatial Coordinates

addxyz xadd yadd zadd is1 thru is2
addxyz xadd yadd zadd is1 is2 is3.....

Cloning (Duplicating) Surfaces

clones ns is1 thru is2
clones ns is1 is2 is3.....

Reduced, Reflecting Geometry

xabove x0
yabove y0
zabove z0

xbelow x0
ybelow y0
zbelow z0

New Sources

source19 (spherical shell), **source 20** (cylindrical shell), and **source21** (a rectangular box) allow you to sample from irregularly shaped zones,

source19 nz1 thru nz2 ri ro [x0 y0 z0]
s19 nz1 thru nz2 ri ro [x0 y0 z0]
s19g nz1 thru nz2 ri ro [x0 y0 z0]

- center of the sphe

source20 nz1 thru nz2 z1 z2 ri r0 [x0 y0]
s20 nz1 thru nz2 z1 z2 ri r0 [x0 y0]
s20g nz1 thru nz2 z1 z2 ri r0 [x0 y0]

source21 nz1 thru nz2 x1 x2 y1 y2 z1 z2
s21 nz1 thru nz2 x1 x2 y1 y2 z1 z2
s21g nz1 thru nz2 x1 x2 y1 y2 z1 z2

Defining Zone Volume by User Input

volume nz1 vz1 nz2 vz2 nz3 vz3…..

Changes in sentinels

"BEST" Options, now set for you

Sentinels 20 (self-shielding), 25 (fluorescence) and 39 (thermal scattering) are now all initialized as ON. You should NEVER use these unless you really do want to turn off this physics.

Sentinels 8, 9, 13, 14, 15 and 16, limit the energy range for tracking and tallying neutrons and photons. These are now automatically set for you, and you should NEVER use these unless you really do want to limit these ranges.

New Sentinels, 54 through 57

54 - definition of the number of neutrons per fission; use either $<\nu>$, or sample the probability distribution of ν.

55 - define a correlated spontaneous fission source.

56 - what to tally with tally types 11 and 12: neutrons entering a zone, absorption, capture, fission….

57 - use either total or prompt $<\nu>$.

References

[1] "**TART95**: A Coupled Neutron-Photon Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-MA-121319, July 1995, by D.E. Cullen, A.L. Edwards and E.F. Plechaty

[2] "**TART96**: A Coupled Neutron-Photon 3-D, Combinatorial Geometry Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-ID-126455, November, 1996, by D.E. Cullen.

[3] "**PREPRO2000**: The 2000 ENDF/B Pre-Processing Codes," The International Atomic Energy Agency, Vienna, Austria, IAEA-NDS-39, Rev. 10, April 2000, by D.E. Cullen.

[4] "**THERMAL**: A Routine Designed to Calculate Neutron Thermal Scattering," Lawrence Livermore National Laboratory, UCRL-ID-120560-Rev-1, Sept. 1995, by D.E. Cullen.

[5] "A Temperature Dependent ENDF/B-VI, **Release 4** Cross Section Library," Lawrence Livermore National Laboratory, UCRL-ID-127776, by D.E. Cullen.

[6] "**EPDL97**: the Evaluated Photon Data Library, '97 Version," Lawrence Livermore National Laboratory, UCRL--50400, Vol. 6, Rev. 5, by D.E. Cullen; this is now the official ENDF/B-VI photon interaction data library.

[7] "Nuclear Cross Section Preparation", by D.E. Cullen, Chapter 1, Volume I, "Handbook of Nuclear Reactions Calculations," editor Yigal Ronon, CRC Press, Boca Raton, Florida (1986).

[8a] "**TART97**: A Coupled Neutron-Photon 3-D, Combinatorial Geometry Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-ID-126455, Rev. 1, November, 1997, by D.E. Cullen

[8b] "**TART98**: A Coupled Neutron-Photon 3-D, Combinatorial Geometry Time Dependent Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-ID-126455, Rev. 2, November, 1998, by D.E. Cullen

[8c] "**TART2000**: A Coupled Neutron-Photon 3-D, Combinatorial Geometry Time Dependent Monte Carlo Transport Code," Lawrence Livermore National Laboratory, UCRL-ID-126455, Rev. 3, November, 2000, by D.E. Cullen

[9] "A Simple Model of Photon Transport", by D.E. Cullen, Nuclear Instrumentation and Methods in Physics Research B101 (1995) pp 499-510. The original extended form of this paper is now available on-line at the TART website, http://reddog1.llnl.gov

[10] "A Temperature Dependent ENDF/B-VI, **Release 7** Cross Section Library," Lawrence Livermore National Laboratory, UCRL-ID-127776, Rev. 1 (2000), by D.E.

Cullen, and "A Temperature Dependent ENDF/B-VI, **Release 8** Cross Section Library," Lawrence Livermore National Laboratory, UCRL-ID-127776, Rev. 2 (2003), by D.E. Cullen.

[11] Private Communication, (1988), by R.J. Howerton

[12] "New Thermal Neutron Scattering Files for ENDF/B-VI Release 2", Los Alamos National Laboratory (March 1994), by R.E. MacFarlane

University of California
Lawrence Livermore National Laboratory
Technical Information Department
Livermore, CA 94551